

IEEE/ACM
2022 INTERNATIONAL
CONFERENCE ON
COMPUTER-AIDED
DESIGN
2022 ICCAD CAD Contest Problem C:
Microarchitecture Design Space Exploration

41st Edition

Sicheng Li^{1,†}, Chen Bai^{1, 2, †}, Xuechao Wei¹, Bizhao Shi¹,
Yen-Kuang Chen¹, Yuan Xie¹

¹Alibaba Group ²The Chinese University of Hong Kong
{sicheng.li, baichen.bai}@alibaba-inc.com

Oct. 30, 2022

† The two authors contributed equally to this work.



① Introduction

② Design Space Exploration

③ Problem Formulation

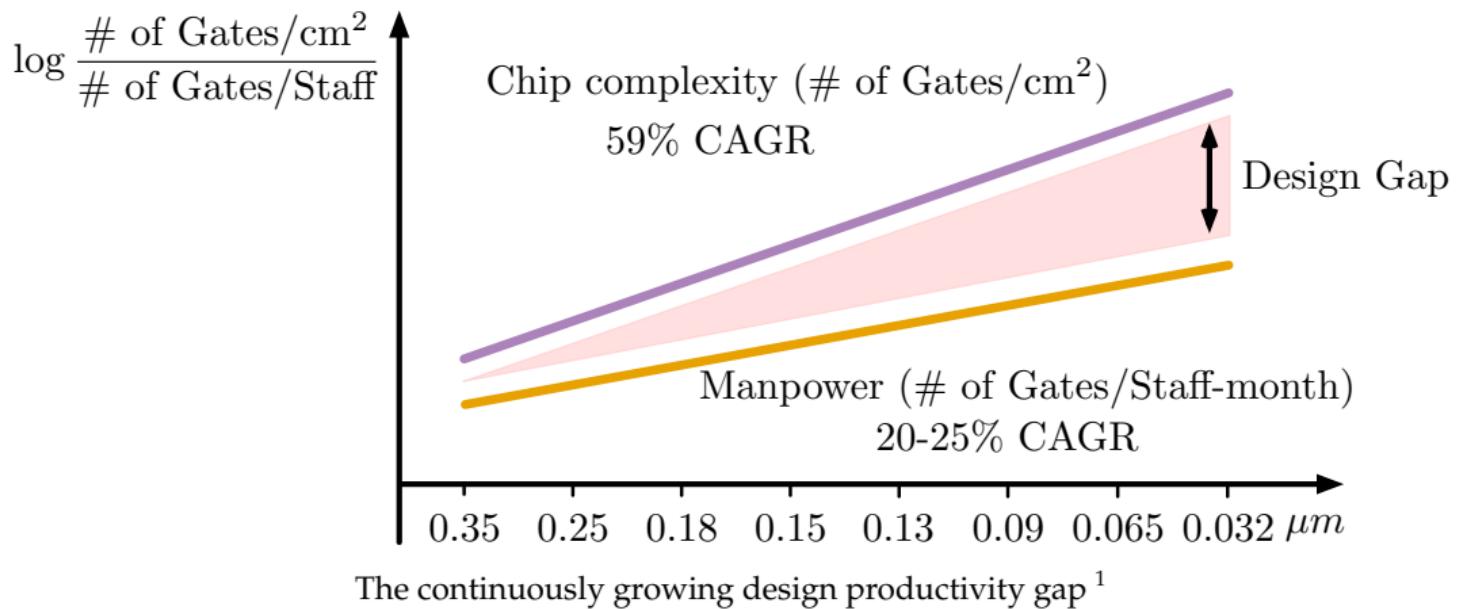
④ Evaluation

⑤ Acknowledgement

Introduction

Introduction

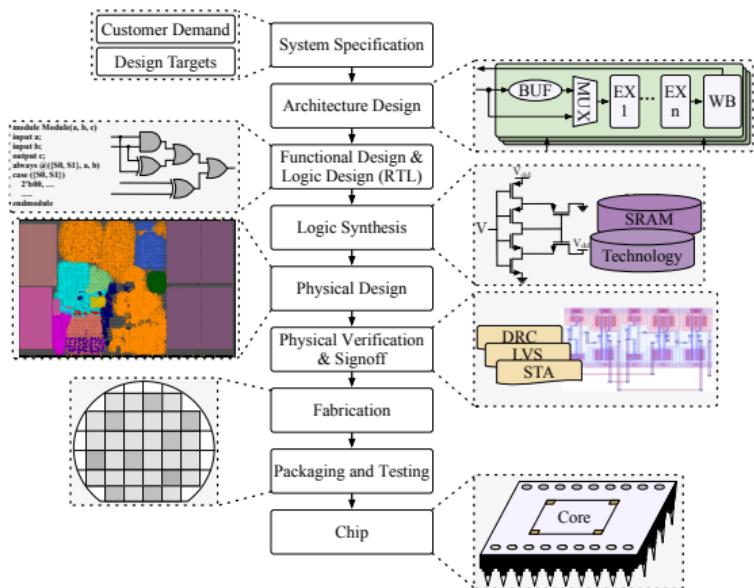
Design Productivity Gap



¹ SEMICO Research Corp. SoC Silicon and Software 2018 Design Cost Analysis: How Rising Costs Impact SoC Design Starts 2018

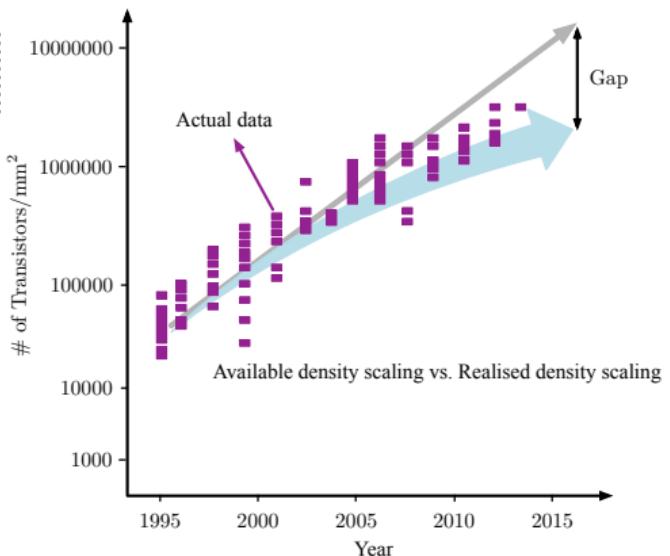
Introduction

Design Quality Gap



Overview of the VLSI Flow

(Tools and flows have steadily increased in complexity)



Design quality is not scaling

- **Incomplete, fabricatable prototypes** over fully featured models
- **Collaborative, flexible teams** over rigid silos
- **Improvement of tools and generators** over improvement of the instance
- **Response to change** over following a plan

Key Points

- Incomplete, fabricatable prototypes → Full RTL implementation model
- Improvement of tools and generators → Parametric chip generator
- Response to change → Available tape-out candidate for any product deadline

Introduction RISC-V Implementations



RISC-V Implementations

- Infrastructure

- Bluespec System Verilog [Nikhil 2004], ROHD², Chisel [Bachrach et al. 2012], PyMTL [Lockhart, Zibrat, and Batten 2014], SpinalHDL³
- Mamba [Jiang, Ilbeyi, and Batten 2018], FireMarshal [Pemberton and Amid 2021], Dromajo [Kabylkas et al. 2021], ChiselVerifier [Dobis et al. 2021]
- Hammer [Wang et al. 2020]
- ...

- Methodology

- BOOM-Explorer [Bai et al. 2021]
- ...

- Platform

- OpenPiton [Balkind et al. 2016], Chipyard [Amid et al. 2020], CHIPKIT [Whatmough et al. 2020]
- ...

²Rapid Open Hardware Development (ROHD) Framework (2021).

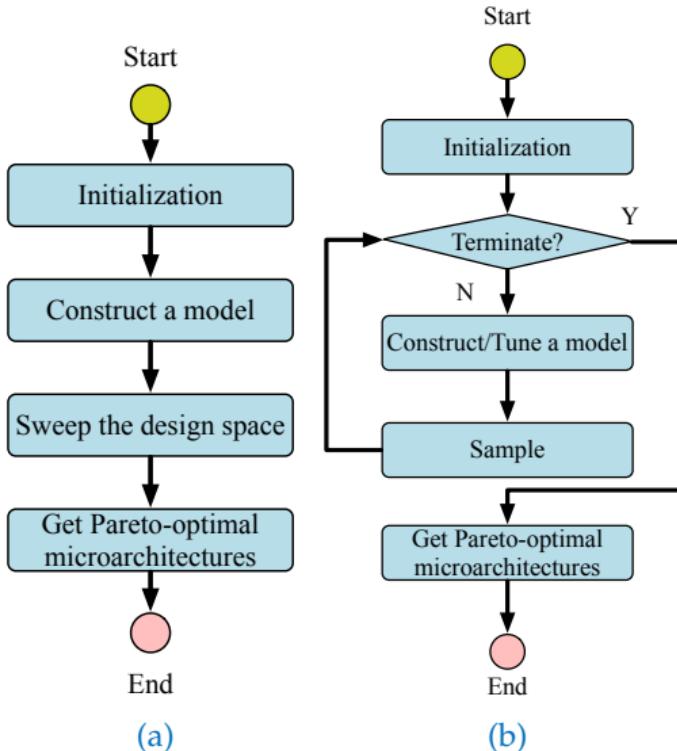
<https://github.com/intel/rohd>

³SpinalHDL: A Language to Describe Digital Hardware (2020).

<https://github.com/SpinalHDL/SpinalHDL>

Design Space Exploration

Design Space Exploration



(a) Offline design space exploration flow. (b) Online design space exploration flow.

Domination Criterion

For an n -objective maximization problem, a vector of objective values \mathbf{y}' is said to dominate \mathbf{y} if

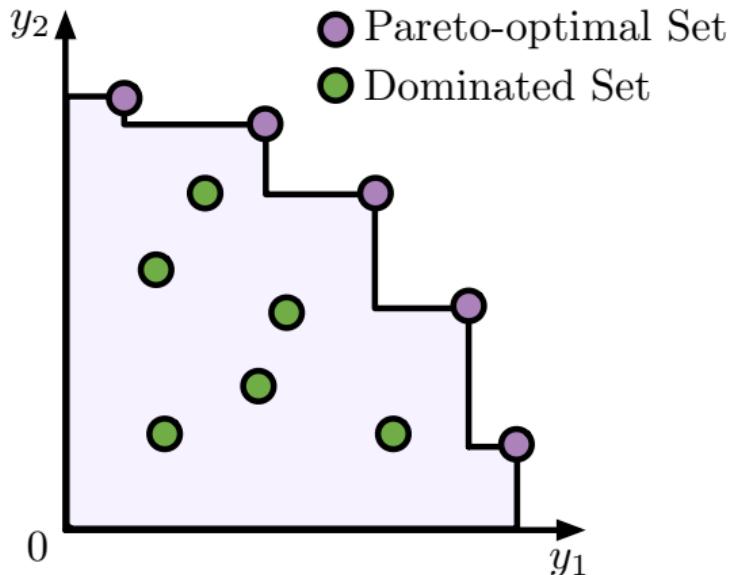
$$\begin{aligned} \forall i \in [1, n], \quad & \mathbf{y}' \geq \mathbf{y}; \\ \exists j \in [1, n], \quad & \mathbf{y}' > \mathbf{y}, \end{aligned} \tag{1}$$

Hence, we denote $\mathbf{y}' \succeq \mathbf{y}$. Otherwise, $\mathbf{y}' \not\succeq \mathbf{y}$.

Pareto-optimal Set

Given $Y = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$, the Pareto-optimal set is to be defined as

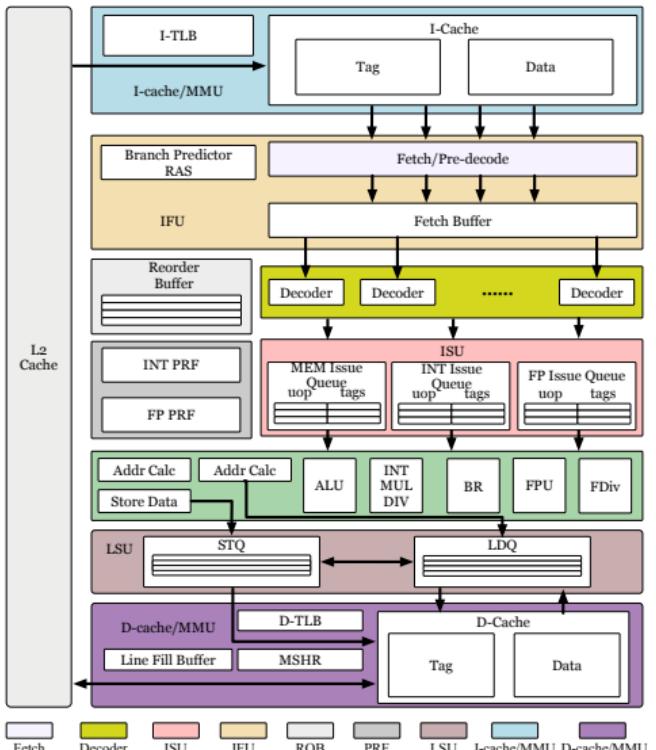
$$\mathcal{P}(Y) = \{\mathbf{y}_i \in Y \mid \mathbf{y}_j \not\succeq \mathbf{y}_i, \forall j \in Y \setminus \{\mathbf{y}_i\}\}.$$



Overview of Pareto-optimal Set

Design Space Exploration

RISC-V BOOM



Overview of RISC-V BOOM [Asanovic, Patterson, and C. Celio 2015; C. P. Celio 2017; Zhao et al. 2020]

Design Space Exploration

RISC-V Microarchitecture Design Space

Table: Microarchitecture Design Space of BOOM

Design	Boxes									Total
	Fetch	Decoder	ISU	IFU	ROB	PRF	LSU	I-cache/MMU	D-cache/MMU	
sub-design-1	1	1	1, 2, 3	1, 2, 3	1, 2, 3, 4	1, 2, 3	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	
sub-design-2	1	2	4, 5, 6	4, 5, 6	5, 6, 7	4, 5, 6	4, 5, 6	1, 2, 3, 4	1, 2, 3, 4	
sub-design-3	2	3	7, 8, 9	7, 8, 9	8, 9, 10	7, 8, 9	7, 8, 9	5, 6, 7	5, 6, 7	15633
sub-design-4	2	4	10, 11, 12	10, 11, 12	11, 12, 13	10, 11, 12	10, 11, 12	5, 6, 7	8, 9, 10	
sub-design-5	2	5	13, 14, 15	12, 13, 14	14, 15, 16	11, 12, 13	10, 11, 12	5, 6, 7	8, 9, 10	

Design Space Exploration

RISC-V Microarchitecture Design Space

Table: Components I

Index	Fetch	Decoder	LSU ¹		I-cache/MMU				D-cache/MMU					
	Width	Width	LDQ	STQ	Sets	Ways	I-TLBSets	I-TLBWays	Sets	Ways	RP ²	MSHR	D-TLBSets	D-TLBWays
1	4	1	8	8	64	4	1	32	64	4	0	2	1	8
2	8	2	6	6	32	8	1	32	64	4	0	2	1	32
3		3	12	12	32	4	1	16	64	6	1	2	1	8
4		4	16	16	64	1	1	16	64	4	1	2	1	32
5		5	12	12	64	8	1	32	64	2	0	4	1	32
6			20	20	64	8	2	16	64	4	1	4	1	32
7			24	24	64	8	2	32	64	4	1	8	1	32
8			22	22	64	8	2	32	64	8	0	8	2	32
9			28	28					64	8	1	8	2	32
10			32	32					64	8	1	8	1	32
11			28	28										
12			36	36										

¹ LDQ and STQ are shared for load and store queue entries, respectively.

² RP is shorted for a replacement policy. Specifically, 0 denotes LRU, and 1 represents Pseudo LRU.

Design Space Exploration

RISC-V Microarchitecture Design Space

Table: Components II

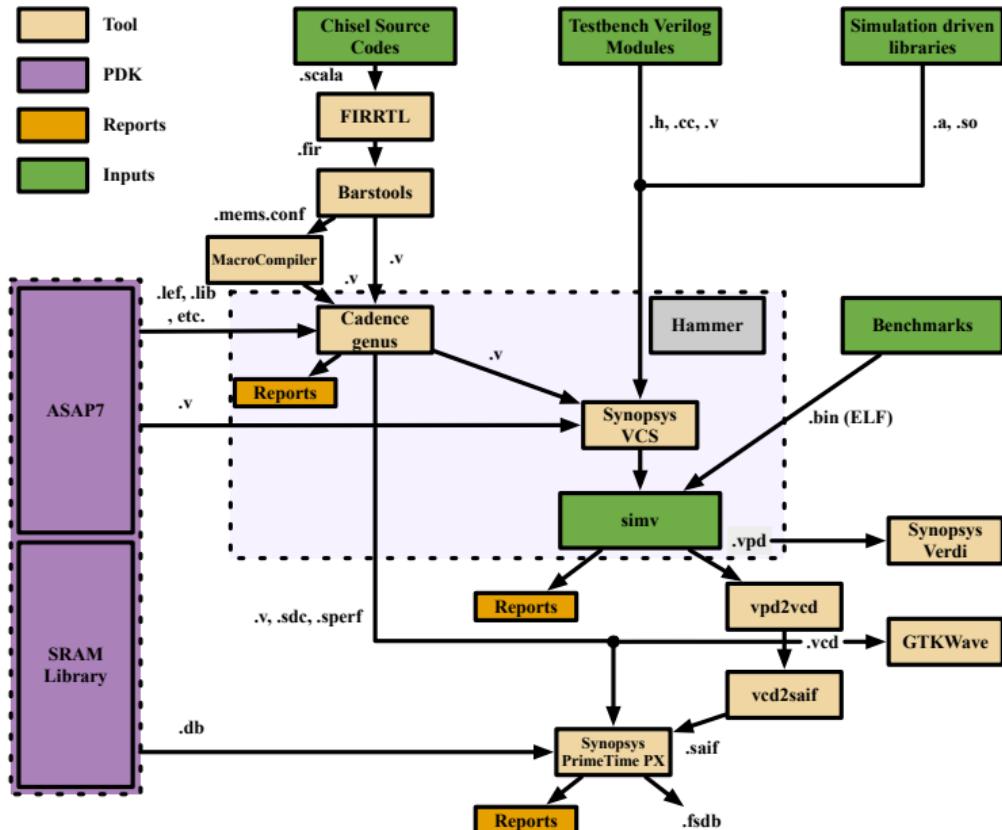
Index	ISU ¹									IFU ²			ROB	PRF ³	
	MEM.DW	MEM.IW	MEM.QE	INT.DW	INT.IW	INT.QE	FP.DW	FP.IW	FP.QE	Tag	FBE	FTQ	Entries	INT	FP
1	1	1	8	1	1	8	1	1	8	8	8	16	32	52	48
2	1	1	6	1	1	6	1	1	6	6	6	14	30	42	38
3	1	1	10	1	1	12	1	1	12	10	12	20	34	62	58
4	2	1	12	2	2	20	2	1	16	12	16	32	36	80	64
5	2	2	12	2	2	20	2	2	16	10	14	30	64	70	54
6	2	2	14	2	2	24	2	2	20	14	20	36	60	90	74
7	3	1	16	3	3	32	3	1	24	16	24	32	72	100	96
8	3	2	16	3	3	32	3	2	24	14	21	30	96	90	86
9	3	2	20	3	3	36	3	3	28	18	30	36	90	110	106
10	4	2	24	4	4	40	4	2	32	20	32	40	108	118	118
11	4	2	28	4	4	44	4	4	36	22	36	44	128	128	128
12	4	2	22	4	4	36	4	2	28	24	40	48	132	138	138
13	5	2	24	5	5	40	5	2	32	20	35	40	136	146	146
14	5	2	26	5	4	44	5	4	36	26	45	50	130		
15	5	2	28	5	5	48	5	5	40				120		
16													140		

¹ DW, IW, and QE are shorted for dispatch width, issue width, and queue entries.

² FBE and FTQ are shorted for fetch buffer entries and fetch target queue entries, respectively.

³ INT and FP are shorted for the number of integer and floating-point physical registers, respectively.

Design Space Exploration BOOM VLSI Flow



Problem Formulation

Our Solution Encoding

Microarchitecture embedding is defined as a combination of candidate values for each component of a microarchitecture, as shown in Table 1, Table 2, and Table 3.

Our Optimization Target

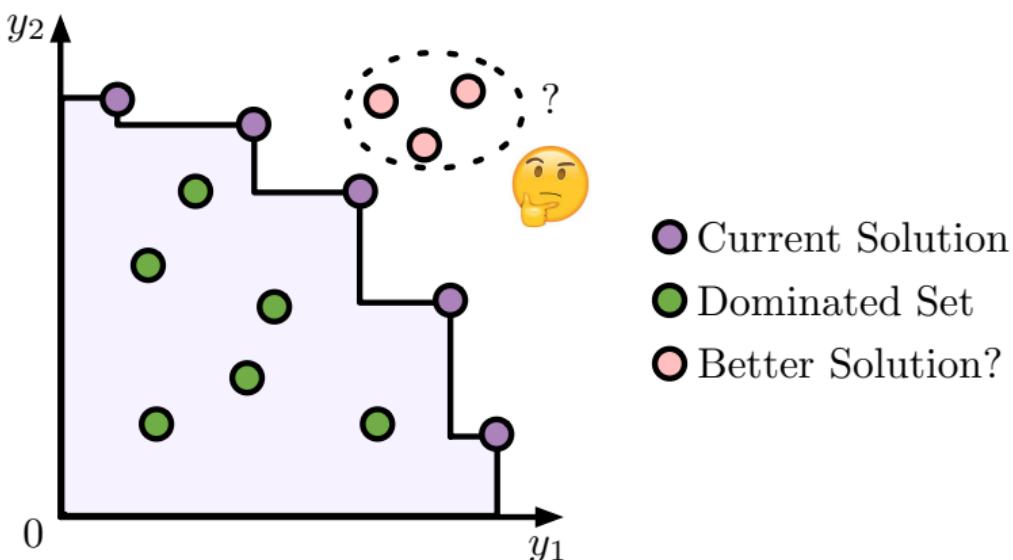
Performance, power, and area (PPA) are three optimization targets.

- Microarchitecture embedding is denoted as a feature vector x .
- A better microarchitecture has higher performance and lower power and area.

Our Problem

Given a design space $\mathcal{D} = \{x_1, x_2, \dots, x_n\}$, experiment f maps the design space \mathcal{D} to PPA metric value space $Y = \{y_1, y_2, \dots, y_n\}$.

The microarchitecture design space exploration is to find $X \subseteq \mathcal{D}$, whose objective values are $\mathcal{P}(Y)$ as far as possible.



Where is the Pareto-optimal Set?

Evaluation

- Dataset of 15,633 RISC-V BOOM microarchitectures
- Contest Benchmarking Platform
- Evaluation Metrics
- Online Ranking Platform

Overview of the dataset, each PPA values of a microarchitecture is obtained from Figure 6

- Get started w. the contest benchmarking platform

```
1 pip3 install iccad-contest
```

- Contest solution implementation: Python3.5+

Listing: Template of the Solution Implementation

```
1   from iccad_contest.abstract_optimizer import
2       AbstractOptimizer
3   from iccad_contest.design_space_exploration import
4       experiment
5
6   class YourAlgorithm(AbstractOptimizer):
7       primary_import = "iccad_contest"
8
9       def __init__(self, design_space):
10          #
11          #      build a wrapper class for an optimizer.
12          #
13          #      parameters
14          #      -----
15
```

15

17

19

21

23

25

27

```
#      design_space: <class "MicroarchitectureDesignSpace">
#
AbstractOptimizer.__init__(self, api_config)
# do whatever other setup is needed
# ...
def suggest(self):
#
#      get a suggestion from the optimizer.
#
#      returns
#
#      -----
#      next_guess: <list> of <list>
#                  list of 'self.nSuggestions'
#      suggestion(s).
```

```
28         #           each suggestion is a microarchitecture
29         #           embedding .
30
31         #
32         # do whatever is needed to get the parallel
33         #   guesses
34         #
35         # ...
36         return x_guess
37
38     def observe(self, x, y):
39         #
40         #           send an observation of a suggestion back
41         #           to the optimizer .
42
43         #
44         #           parameters
45         #
46         -----
47         #           x: <list> of <list>
48         #           the output of 'suggest'.
```

```
43     #      y: <list> of <list>
43     #              corresponding values where each 'x' is
43     # mapped to .
44
45     # update the model with new objective function
45     # observations
46     # ...
47     # no return statement needed
48
49 if __name__ == "__main__":
50     # this is the entry point for experiments , so pass
50     # the class to
51     # 'experiment_main_entry' to use this optimizer .
51     # this statement must be included in the wrapper
51     # class file :
52
53     experiment(YourAlgorithm)
```

Listing: Data structure Definition of the Design Space

```
descriptions = {  
    2     "sub-design-1": {  
        4         "Fetch": [1],  
        4         "Decoder": [1],  
        4         "ISU": [1, 2, 3],  
        6         ...  
    8     }  
  
10 components_mappings = {  
    12     "Fetch": {  
        12         "description": ["FetchWidth"],  
        12         "1": [4]  
    14     },  
    14     "Decoder": {  
        14         "description": ["DecoderWidth"],  
        14         "1": [4]  
    16     }  
}
```

```
16          "description": ["DecodeWidth"] ,  
17          "1": [1]  
18      },  
19      "ISU": {  
20          "description": [  
21              "MEM_INST.DispatchWidth", "  
22                  MEM_INST.IssueWidth"  
23              "MEM_INST.NumEntries", "  
24                  INT_INST.DispatchWidth",  
25                  "INT_INST.IssueWidth", "  
26                  INT_INST.NumEntries",  
27                  "FP_INST.DispatchWidth", "  
28                  FP_INST.IssueWidth",  
29                  "FP_INST.NumEntries"  
30          ],  
31          "1": [1, 1, 8, 1, 1, 8, 1, 1, 8],  
32          "2": [1, 1, 6, 1, 1, 6, 1, 1, 6],
```

```
30                     "3": [1, 1, 10, 1, 1, 12, 1, 1, 12]
31     },
32     "IFU": {
33         "description": ["BranchTag", "FetchBufferEntries", "FetchTargetQueue"]
34         "1": [8, 8, 16],
35         ...
36     },
37     ...
38 }
```

[Listing:](#) Benchmarking Platform Help Menu

```
1 $ python3 random-search-optimizer.py -h

3 usage: random-search-optimizer.py [-h] [-o OUTPUT_PATH]
        [-u UUID] [-s SOLUTION_SETTINGS] [-q NUM_OF_QUERIES
        ]

5 ICCAD'22 Contest Platform - solutions evaluation

7 optional arguments:
9     -h, --help            show this help message and exit
     -o OUTPUT_PATH, --output-path OUTPUT_PATH contest
                           output path specification
     -u UUID, --uuid UUID  universally unique identifier (UUID)
                           specification
```

```
11 -s SOLUTION_SETTINGS, --solution-settings  
    SOLUTION_SETTINGS solution submission  
    specification  
-q NUM_OF_QUERIES, --num-of-queries NUM_OF_QUERIES  
    the number of queries specification
```

Listing: Example Command of Benchmarking Platform

```
$ python3 random-search-optimizer.py -o output -u "00
ef538e88634ddd9810d034b748c24d" -q 20
2 ...
[INFO]: summary for the solution, the best Pareto
hypervolume: 37.59654046710655, the best Pareto
hypervolume difference: 66.67984662813456 cost:
164902.3422778734.
4 ...

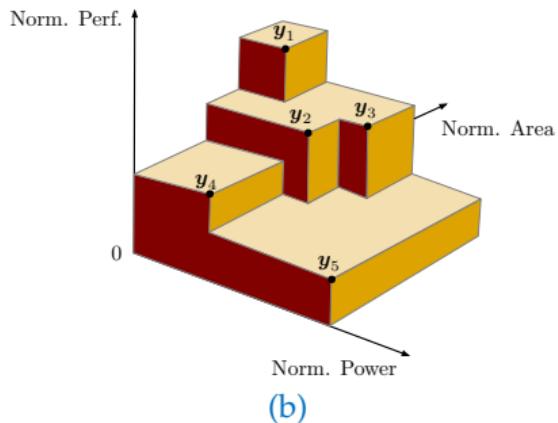
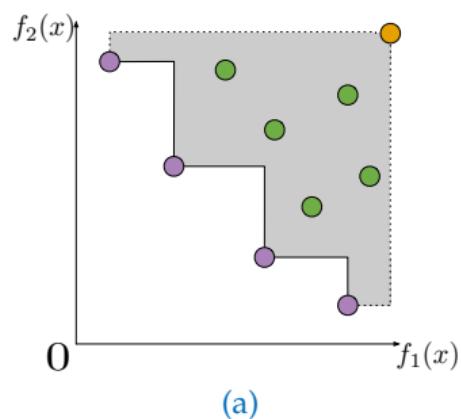
---


```

- Pareto hypervolume
- Overall running time: the total time of algorithms including VLSI flow runtime cost.
- A final score is calculated based on Pareto hypervolume and overall running time

Pareto hypervolume [Bai et al. 2021]

$$\text{PVol}_{v_{\text{ref}}}(\mathcal{P}(Y)) = \int_Y \mathbb{1}[\mathbf{y} \not\geq \mathbf{v}_{\text{ref}}] [1 - \prod_{\mathbf{y}_* \in \mathcal{P}(Y)} \mathbb{1}[\mathbf{y}_* \not\geq \mathbf{y}]] d\mathbf{y}, \quad (2)$$



(a). An example overview of the Pareto hypervolume in the two dimensional space (b). An example overview of the Pareto hypervolume in the three dimensional space, *i.e.*, power, performance, and area.

Final Score

$$\text{score} = \text{PVol}_{v_{\text{ref}}} \cdot \begin{cases} \alpha - \frac{\text{ORT} - \theta}{\theta}, & \text{ORT} \geq \theta \\ \alpha + \left| \frac{\text{ORT} - \theta}{\theta} \right|, & \text{ORT} < \theta \end{cases}, \quad (3)$$

α is an ORT score baseline, equal to 6, and θ is a pre-defined ORT budget, equivalent to 2625000.

Evaluation Online Ranking Platform



ICCAD 2022 CAD Contest

Problem C

Introduction [last update 7/30](#)



Modern chip development requires high cost in design time and workforce. The reason for the expensive chip development cycle lies in two folds. On the one hand, the pre-defined performance, power, and area (PPA) targets of the chip are set aggressively, hoping to deliver the following generation products comparable and superior to market competitors. On the other hand, the design complexity of the chip (number of gates / chip area) is continuously increasing, leaving the improvement in design capability (number of gates / staff × month) behind a considerable margin, causing an imbalance between the required design efforts and the cost of input. Thanks to the agile design paradigm provided by advanced hardware description languages, e.g., Chisel, and flexible and parameterized hardware generators, chip architects and engineers possess the capability to deliver a high-quality chip within a limited time budget. To further enhance the chip design ability built on top of the agile development paradigm for industry, exploring a series of chips in a given design space to achieve different degrees of trade-offs w.r.t. performance, power, and area in a short time is necessary. We look for effective and practical design space exploration algorithms to solve the problem. Specifically, in this contest problem, we focus on microarchitecture exploration of processors, i.e., central processing units (CPU).

[Problem details](#)

Instruction [last update 8/1](#)

ICCAD Contest Platform is the platform to benchmark your submissions. All codebase is based on Python programming language. Please follow the instruction to implement your answers.
Follow the [ICCAD Contest Starter Kit](#) to prepare your submissions for upload.

Submission [last update 7/25](#)

[Visit the page](#) to submit your answer.

The initial submission user name and password are the same as registering your accounts in the contest. You can change your password when you log in to the submission page.

Ranking [last update 8/7](#)

[Visit the page](#) to check the latest ranking. The following submission deadline for the ranking is 8/10, 17:00:00 (GTM+8), 2022. The ranking list will be updated accordingly.

Miscellaneous information

Please visit the [ICCAD Contest](#) for more information.

Copyright © 2022 CAD Contest. All Rights Reserved.



Overview of the online ranking platform.



ICCAD 2022

CAD Contest



Problem C Ranking last update 9/04 (answers submitted before 9/02)



Team	Pareto Hypervolume	Overall Running Time (ORT)	Score	Description
cadc1021	7.90015910	1086890.06081569	52.03027006	-
cadc1013	7.68893083	940331.78222627	51.07481625	-
cadc1012	7.78248535	1183971.61405234	50.96721015	-
cadc1004	7.79817878	1438698.36862783	50.31326021	-
cadc1001	7.68703481	1204461.50604989	50.28213483	-
cadc0008	7.82024213	1532006.99497922	50.17783182	-
cadc1018	7.75015927	1530718.41317414	49.73175811	-
cadc1028	7.79705446	1750357.90583499	49.38028184	-
cadc1006	7.50406694	3119253.84579519	43.61148204	-
cadc1023	N/A	N/A	N/A	[WARN]: failed in evaluating microarchitecture embeddings. set the evaluation of the limit.



Copyright © 2022 CAD Contest. All Rights Reserved.

Ranking list

- The online ranking platform will be maintained after the contest.
- Everybody can register and participate in solving the problem.
- Evaluation according to fixed frequency.
- Address: <http://47.93.191.38/>

Acknowledgement

Acknowledgement

We would like to thank

- **Prof. Yuzhe Ma**, from the Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou)
- **Prof. Bei Yu**, from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong SAR.

They suggested valuable ideas and helped to review the problem.

THANK YOU!

- Alon Amid et al. (2020). “Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs”. In: *IEEE Micro* 40.4, pp. 10–21.
- Krste Asanovic, David A Patterson, and Christopher Celio (2015). *The Berkeley Out-of-order Machine (BOOM): An Industry-competitive, Synthesizable, Parameterized RISC-V Processor*. Tech. rep. University of California at Berkeley.
- Jonathan Bachrach et al. (2012). “Chisel: Constructing Hardware in a Scala Embedded Language”. In: *ACM/IEEE Design Automation Conference (DAC)*, pp. 1212–1221.
- Chen Bai et al. (2021). “BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework”. In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, pp. 1–9.
- Jonathan Balkind et al. (2016). “OpenPiton: An Open Source Manycore Research Framework”. In: *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* 51.4, pp. 217–232.
- Christopher Patrick Celio (2017). *A Highly Productive Implementation of an Out-of-Order Processor Generator*. eScholarship, University of California.

- Andrew Dobis et al. (2021). “ChiselVerify: An Open-source Hardware Verification Library for Chisel and Scala”. In: *IEEE Nordic Circuits and Systems Conference (NorCAS)*. IEEE, pp. 1–7.
- Shunning Jiang, Berkin Ilbeyi, and Christopher Batten (2018). “Mamba: Closing the Performance Gap in Productive Hardware Development Frameworks”. In: *ACM/IEEE Design Automation Conference (DAC)*. IEEE, pp. 1–6.
- Nursultan Kabylkas et al. (2021). “Effective Processor Verification with Logic Fuzzer Enhanced Co-simulation”. In: *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 667–678.
- Yunsup Lee et al. (2016). “An Agile Approach to Building RISC-V Microprocessors”. In: *IEEE Micro* 36.2, pp. 8–20.
- Derek Lockhart, Gary Zibrat, and Christopher Batten (2014). “PyMTL: A Unified Framework for Vertically Integrated Computer Architecture Research”. In: *IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, pp. 280–292.

- Rishiyur Nikhil (2004). “Bluespec System Verilog: Efficient, Correct RTL from High Level Specifications”. In: *ACM/IEEE International Conference on Formal Methods and Models for Co-Design (MEMCOD)*. IEEE, pp. 69–70.
- Nathan Pemberton and Alon Amid (2021). “Firemarshal: Making HW/SW Co-design Reproducible and Reliable”. In: IEEE, pp. 299–309.
- Rapid Open Hardware Development (ROHD) Framework* (2021).
<https://github.com/intel/rohd>.
- SEMICO Research Corp. SoC Silicon and Software 2018 Design Cost Analysis: How Rising Costs Impact SoC Design Starts* (2018).
<https://semico.com/content/soc-silicon-and-software-2018-design-cost-analysis-how-rising-costs-impact-soc-design-starts>.
- SpinalHDL: A Language to Describe Digital Hardware* (2020).
<https://github.com/SpinalHDL/SpinalHDL>.
- Edward Wang et al. (2020). “A Methodology for Reusable Physical Design”. In: *IEEE International Symposium on Quality Electronic Design (ISQED)*. IEEE, pp. 243–249.

- Paul N Whatmough et al. (2020). “CHIPKIT: An Agile, Reusable Open-source Framework for Rapid Test Chip development”. In: *IEEE Micro* 40.4, pp. 32–40.
- Jerry Zhao et al. (2020). “SonicBOOM: The 3rd Generation Berkeley Out-of-order Machine”. In: *Fourth Workshop on Computer Architecture Research with RISC-V*. Vol. 5.